In the Claims

Amend claim 9 as follows:

1. (original) A method of parallel processing comprising:

providing a first thread which represents an independent flow of control managed by a program structure, said first thread having two states, a first state processing work for the program structure and a second state undispatched awaiting work to process;

providing a second thread which represents an independent flow of control managed by a program structure separate from the first thread;

using the second thread to prepare work for the first thread to process;

placing the work prepared by the second thread in a queue for processing by the first thread;

if the first thread is awaiting work to process when the work prepared by the second thread is placed in the queue, dispatching the first thread and using it to process the work in the queue;

if the first thread is processing other work when the work prepared by the second thread is placed in the queue, using the first thread to complete processing of the other work, access the work in the queue, and then process the work in the queue.

2. (original) The method of claim 1 wherein the second thread continues to place additional work in the queue, and the first thread sequentially processes the additional work in the queue as it completes processing prior work.

3. (original) The method of claim 1 wherein the second thread marks the work placed in the first thread queue as not complete.

4. (original) The method of claim 1 wherein if the first thread is processing other work when the work prepared by the second thread is placed in the queue, and when the first thread completes processing of the work in the queue, using the first thread to mark the completed work as complete, wherein subsequent work from the second thread is made to wait until the previous work in the first thread is marked complete.

A1

5. (original) The method of claim 1 wherein the first thread is reused to process other work.

6. (original) The method of claim 4 wherein the program structure destroys the first thread after it completes a desired amount of work.

7. (original) A method of parallel processing comprising:

providing a first thread which represents an independent flow of control managed by a program structure, said first thread having two states, a first state processing work for the program structure and a second state undispatched awaiting work to process;

providing a second thread which represents an independent flow of control managed by a program structure separate from the first thread;

using the second thread to prepare work for the first thread to process;

placing the work prepared by the second thread in a queue for processing by the first thread, the work placed in the first thread queue being marked as not complete;

if the first thread is awaiting work to process when the work prepared by the second thread is placed in the queue, dispatching the first thread and using it to process the work in the queue;

if the first thread is processing other work when the work prepared by the second thread is placed in the queue, using the first thread to complete processing of the other work, access the work in the queue, and then process the work in the queue;

using the second thread to place additional work in the queue; and

using the first thread to sequentially process the additional work in the queue as it completes processing prior work.

8. (original) The method of claim 7 wherein if the first thread is processing other work when the work prepared by the second thread is placed in the queue, and when the first thread completes processing of the work in the queue, using the first thread to mark the completed work as complete, wherein subsequent work from the second thread is made to wait until the previous work in the first thread is marked complete.

9. (currently amended) The method of claim 7 wherein the first thread is reused to process other work.

10. (original) The method of claim 8 wherein the program structure destroys the first thread after it completes a desired amount of work.

11. (original) A program storage device readable by a machine, tangibly embodying a program of instructions executable by the machine to perform method steps of parallel processing using i) a first thread which represents an independent flow of control managed by a program structure, said first thread having two states, a first state processing work for the program structure and a second state undispatched awaiting work to process, and ii) a second thread which represents an independent flow of control managed by a program structure separate from the first thread, said method steps comprising:

using the second thread to prepare work for the first thread to process;

placing the work prepared by the second thread in a queue for processing by the first thread;

if the first thread is awaiting work to process when the work prepared by the second thread is placed in the queue, dispatching the first thread and using it to process the work in the queue;

if the first thread is processing other work when the work prepared by the second thread is placed in the queue, using the first thread to complete processing of the other work, access the work in the queue, and then process the work in the queue.

12. (original) The program storage device of claim 11 wherein, in the method steps, the second thread continues to place additional work in the queue, and the first

thread sequentially processes the additional work in the queue as it completes processing prior work.

13. (original) The program storage device of claim 11 wherein, in the method steps, the second thread marks the work placed in the first thread queue as not complete.

14. (original) The program storage device of claim 11 wherein, in the method steps, if the first thread is processing other work when the work prepared by the second thread is placed in the queue, and when the first thread completes processing of the work in the queue, using the first thread to mark the completed work as complete, wherein subsequent work from the second thread is made to wait until the previous work in the first thread is marked complete.

15. (original) The program storage device of claim 11 wherein, in the method steps, the first thread is reused to process other work.

16. (original) The program storage device of claim 14 wherein, in the method steps, the program structure destroys the first thread after it completes a desired amount of work.